Foolbox

Release 3.3.1

Jonas Rauber

Feb 23, 2021

USER API

Foolbox is a Python toolbox to create adversarial examples that fool neural networks. *Foolbox 3.0* a.k.a. *Foolbox Native* has been completely rewritten from scratch. It is now built on top of EagerPy and comes with native support for these frameworks:

- PyTorch
- TensorFlow
- JAX

Foolbox comes with a *large collection of adversarial attacks*, both gradient-based white-box attacks as well as decision-based and score-based black-box attacks.

The source code and a minimal working example can be found on GitHub.

CHAPTER

ONE

FOOLBOX.MODELS

1.1 Models

Model
PyTorchModel
TensorFlowModel
JAXModel
NumPyModel

1.2 Wrappers

TransformBoundsWrapper

1.3 Detailed description

class foolbox.models.**Model**

transform_bounds(*bounds*)

Returns a new model with the desired bounds and updates the preprocessing accordingly

Parameters **Tuple[float, float]] bounds** (*Union[*foolbox.types.Bounds,) –

Return type foolbox.models.base.Model

class foolbox.models.**PyTorchModel**(*model*, *bounds*, *device=None*, *preprocessing=None*)

Parameters

- **model** (*Any*) –
- **Tuple[float, float]] bounds** (*Union[*foolbox.types.Bounds,) –
- **device** (*Any*) –
- **Any]] preprocessing** (*Optional[Dict[str,*) –

class foolbox.models.**TensorFlowModel**(*model*, *bounds*, *device=None*, *preprocessing=None*)

Parameters

- **model** (*Any*) –
- **Tuple[float, float]] bounds** (*Union[foolbox.types.Bounds,*) –
- **device** (*Any*) –
- **Any]] preprocessing** (*Optional[Dict[str,*) –

class foolbox.models.**JAXModel**(*model*, *bounds*, *preprocessing=None*, *data_format='channels_last'*)

Parameters

- **model** (*Any*) –
- **Tuple[float, float]] bounds** (*Union[foolbox.types.Bounds,*) –
- **Any]] preprocessing** (*Optional[Dict[str,*) –
- **data_format** (*Optional[str]*) –

class foolbox.models.**NumPyModel**(*model*, *bounds*, *data_format=None*)

Parameters

- **model** (*Callable*) –
- **Tuple[float, float]] bounds** (*Union[foolbox.types.Bounds,*) –
- **data_format** (*Optional[str]*) –

class foolbox.models.**TransformBoundsWrapper**(*model*, *bounds*)

Parameters

- **model** (*foolbox.models.base.Model*) –
- **Tuple[float, float]] bounds** (*Union[foolbox.types.Bounds,*) –

transform_bounds(*bounds*, *inplace=False*)

Returns a new model with the desired bounds and updates the preprocessing accordingly

Parameters

- **Tuple[float, float]] bounds** (*Union[foolbox.types.Bounds,*) –
- **inplace** (*bool*) –

Return type foolbox.models.base.Model

CHAPTER

TWO

FOOLBOX.ATTACKS

L2ContrastReductionAttack	Reduces the contrast of the input using a perturbation of the given size
VirtualAdversarialAttack	Second-order gradient-based attack on the logits.
DDNAttack	The Decoupled Direction and Norm L2 adversarial attack.
L2ProjectedGradientDescentAttack	L2 Projected Gradient Descent
LinfProjectedGradientDescentAttack	Linf Projected Gradient Descent
L2BasicIterativeAttack	L2 Basic Iterative Method
LinfBasicIterativeAttack	L-infinity Basic Iterative Method
L2FastGradientAttack	Fast Gradient Method (FGM)
LinfFastGradientAttack	Fast Gradient Sign Method (FGSM)
L2AdditiveGaussianNoiseAttack	Samples Gaussian noise with a fixed L2 size.
L2AdditiveUniformNoiseAttack	Samples uniform noise with a fixed L2 size.
L2ClippingAwareAdditiveGaussianNoiseAttack	Samples Gaussian noise with a fixed L2 size after clipping.
L2ClippingAwareAdditiveUniformNoiseAttack	Samples uniform noise with a fixed L2 size after clipping.
LinfAdditiveUniformNoiseAttack	Samples uniform noise with a fixed L-infinity size
L2RepeatedAdditiveGaussianNoiseAttack	Repeatedly samples Gaussian noise with a fixed L2 size.
L2RepeatedAdditiveUniformNoiseAttack	Repeatedly samples uniform noise with a fixed L2 size.
L2ClippingAwareRepeatedAdditiveGaussianNoiseAttack	Repeatedly samples Gaussian noise with a fixed L2 size after clipping.
L2ClippingAwareRepeatedAdditiveUniformNoiseAttack	Repeatedly samples uniform noise with a fixed L2 size after clipping.
LinfRepeatedAdditiveUniformNoiseAttack	Repeatedly samples uniform noise with a fixed L-infinity size.
InversionAttack	Creates "negative images" by inverting the pixel values.
BinarySearchContrastReductionAttack	Reduces the contrast of the input using a binary search to find the smallest adversarial perturbation
LinearSearchContrastReductionAttack	Reduces the contrast of the input using a linear search to find the smallest adversarial perturbation
L2CarliniWagnerAttack	Implementation of the Carlini & Wagner L2 Attack.
NewtonFoolAttack	Implementation of the NewtonFool Attack.
EADAttack	Implementation of the EAD Attack with EN Decision Rule.
GaussianBlurAttack	Blurs the inputs using a Gaussian filter with linearly increasing standard deviation.
L2DeepFoolAttack	A simple and fast gradient-based adversarial attack.

Continued on next page

Table 1 – continued from previous page

LinfDeepFoolAttack	A simple and fast gradient-based adversarial attack.
SaltAndPepperNoiseAttack	Increases the amount of salt and pepper noise until the input is misclassified.
LinearSearchBlendedUniformNoiseAttack	Blends the input with a uniform noise input until it is misclassified.
BinarizationRefinementAttack	For models that preprocess their inputs by binarizing the inputs, this attack can improve adversarials found by other attacks.
DatasetAttack	Draws randomly from the given dataset until adversarial examples for all inputs have been found.
BoundaryAttack	A powerful adversarial attack that requires neither gradients nor probabilities.
L0BrendelBethgeAttack	L0 variant of the Brendel & Bethge adversarial attack.
L1BrendelBethgeAttack	L1 variant of the Brendel & Bethge adversarial attack.
L2BrendelBethgeAttack	L2 variant of the Brendel & Bethge adversarial attack.
LinfinityBrendelBethgeAttack	L-infinity variant of the Brendel & Bethge adversarial attack.
FGM	alias of `foolbox.attacks.fast_gradient_method.L2FastGradientAttack`
FGSM	alias of `foolbox.attacks.fast_gradient_method.LinfFastGradientAttack`
L2PGD	alias of `foolbox.attacks.projected_gradient_descent.L2ProjectedGradientDescentAttack`
LinfPGD	alias of `foolbox.attacks.projected_gradient_descent.LinfProjectedGradientDescentAttack`
PGD	alias of `foolbox.attacks.projected_gradient_descent.LinfProjectedGradientDescentAttack`

class foolbox.attacks.**L2ContrastReductionAttack**(*, *target=0.5*)

Reduces the contrast of the input using a perturbation of the given size

Parameters **target** (*float*) – Target relative to the bounds from 0 (min) to 1 (max) towards which the contrast is reduced

class foolbox.attacks.**VirtualAdversarialAttack**(*steps*, *xi=1e-06*)

Second-order gradient-based attack on the logits.[1] The attack calculate an untargeted adversarial perturbation by performing a approximated second order optimization step on the KL divergence between the unperturbed predictions and the predictions for the adversarial perturbation. This attack was originally introduced as the Virtual Adversarial Training[1] method.

Parameters

- **steps** (*int*) – Number of update steps.
- **xi** (*float*) – L2 distance between original image and first adversarial proposal.

[1] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, Shin Ishii, "Distributional Smoothing with Virtual Adversarial Training", https://arxiv.org/abs/1507.00677

References

class foolbox.attacks.**DDNAttack**(*, *init_epsilon=1.0*, *steps=10*, *gamma=0.05*)
The Decoupled Direction and Norm L2 adversarial attack.[2]

Parameters

- **init_epsilon** (*float*) – Initial value for the norm/epsilon ball.
- **steps** (*int*) – Number of steps for the optimization.
- **gamma** (*float*) – Factor by which the norm will be modified: new_norm = norm * (1 + or - gamma).

References

class foolbox.attacks.**L2ProjectedGradientDescentAttack**(*, *rel_stepsize=0.025*, *abs_stepsize=None*, *steps=50*, *random_start=True*)
L2 Projected Gradient Descent

Parameters

- **rel_stepsize** (*float*) – Stepsize relative to epsilon
- **abs_stepsize** (*Optional[float]*) – If given, it takes precedence over rel_stepsize.
- **steps** (*int*) – Number of update steps to perform.
- **random_start** (*bool*) – Whether the perturbation is initialized randomly or starts at zero.

class foolbox.attacks.**LinfProjectedGradientDescentAttack**(*, *rel_stepsize=0.03333333333333333*, *abs_stepsize=None*, *steps=40*, *random_start=True*)
Linf Projected Gradient Descent

Parameters

- **rel_stepsize** (*float*) – Stepsize relative to epsilon (defaults to 0.01 / 0.3).
- **abs_stepsize** (*Optional[float]*) – If given, it takes precedence over rel_stepsize.
- **steps** (*int*) – Number of update steps to perform.
- **random_start** (*bool*) – Whether the perturbation is initialized randomly or starts at zero.

class foolbox.attacks.**L2BasicIterativeAttack**(*, *rel_stepsize=0.2*, *abs_stepsize=None*, *steps=10*, *random_start=False*)
L2 Basic Iterative Method

Parameters

- **rel_stepsize** (*float*) – Stepsize relative to epsilon.
- **abs_stepsize** (*Optional[float]*) – If given, it takes precedence over rel_stepsize.
- **steps** (*int*) – Number of update steps.

[2] Jérôme Rony, Luiz G. Hafemann, Luiz S. Oliveira, Ismail Ben Ayed, Robert Sabourin, Eric Granger, "Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses", https://arxiv.org/abs/1811.09600

- **random_start** (*bool*) – Controls whether to randomly start within allowed epsilon ball.

class foolbox.attacks.**LinfBasicIterativeAttack**(*, *rel_stepsize=0.2*, *abs_stepsize=None*, *steps=10*, *random_start=False*)

L-infinity Basic Iterative Method

Parameters

- **rel_stepsize** (*float*) – Stepsize relative to epsilon.
- **abs_stepsize** (*Optional[float]*) – If given, it takes precedence over rel_stepsize.
- **steps** (*int*) – Number of update steps.
- **random_start** (*bool*) – Controls whether to randomly start within allowed epsilon ball.

class foolbox.attacks.**L2FastGradientAttack**(*, *random_start=False*)

Fast Gradient Method (FGM)

Parameters **random_start** (*bool*) – Controls whether to randomly start within allowed epsilon ball.

class foolbox.attacks.**LinfFastGradientAttack**(*, *random_start=False*)

Fast Gradient Sign Method (FGSM)

Parameters **random_start** (*bool*) – Controls whether to randomly start within allowed epsilon ball.

class foolbox.attacks.**L2AdditiveGaussianNoiseAttack**

Samples Gaussian noise with a fixed L2 size.

class foolbox.attacks.**L2AdditiveUniformNoiseAttack**

Samples uniform noise with a fixed L2 size.

class foolbox.attacks.**L2ClippingAwareAdditiveGaussianNoiseAttack**

Samples Gaussian noise with a fixed L2 size after clipping.

The implementation is based on **[#Rauber20]_**.

References

class foolbox.attacks.**L2ClippingAwareAdditiveUniformNoiseAttack**

Samples uniform noise with a fixed L2 size after clipping.

The implementation is based on **[#Rauber20]_**.

References

class foolbox.attacks.**LinfAdditiveUniformNoiseAttack**

Samples uniform noise with a fixed L-infinity size

class foolbox.attacks.**L2RepeatedAdditiveGaussianNoiseAttack**(*, *repeats=100*, *check_trivial=True*)

Repeatedly samples Gaussian noise with a fixed L2 size.

Parameters

- **repeats** (*int*) – How often to sample random noise.
- **check_trivial** (*bool*) – Check whether original sample is already adversarial.

class foolbox.attacks.**L2RepeatedAdditiveUniformNoiseAttack**(*, *repeats=100*, *check_trivial=True*)

Repeatedly samples uniform noise with a fixed L2 size.

Parameters

- **repeats** (*int*) – How often to sample random noise.
- **check_trivial** (*bool*) – Check whether original sample is already adversarial.

class foolbox.attacks.**L2ClippingAwareRepeatedAdditiveGaussianNoiseAttack**(*, *repeats=100*, *check_trivial=True*)

Repeatedly samples Gaussian noise with a fixed L2 size after clipping.

The implementation is based on **[#Rauber20]_**.

References

Parameters

- **repeats** (*int*) – How often to sample random noise.
- **check_trivial** (*bool*) – Check whether original sample is already adversarial.

class foolbox.attacks.**L2ClippingAwareRepeatedAdditiveUniformNoiseAttack**(*, *repeats=100*, *check_trivial=True*)

Repeatedly samples uniform noise with a fixed L2 size after clipping.

The implementation is based on **[#Rauber20]_**.

References

Parameters

- **repeats** (*int*) – How often to sample random noise.
- **check_trivial** (*bool*) – Check whether original sample is already adversarial.

class foolbox.attacks.**LinfRepeatedAdditiveUniformNoiseAttack**(*, *repeats=100*, *check_trivial=True*)

Repeatedly samples uniform noise with a fixed L-infinity size.

Parameters

- **repeats** (*int*) – How often to sample random noise.
- **check_trivial** (*bool*) – Check whether original sample is already adversarial.

class foolbox.attacks.**InversionAttack**(*, *distance=None*)

Creates "negative images" by inverting the pixel values.[7]

[7] Hossein Hosseini, Baicen Xiao, Mayoore Jaiswal, Radha Poovendran, "On the Limitation of Convolutional Neural Networks in Recognizing Negative Images", https://arxiv.org/abs/1607.02533

References

Parameters **distance** (*Optional[foolbox.distances.Distance]*) –

class foolbox.attacks.**BinarySearchContrastReductionAttack**(*, *distance=None*, *binary_search_steps=15*, *target=0.5*)

Reduces the contrast of the input using a binary search to find the smallest adversarial perturbation

Parameters

- **distance** (*Optional[foolbox.distances.Distance]*) – Distance measure for which minimal adversarial examples are searched.
- **binary_search_steps** (*int*) – Number of iterations in the binary search. This controls the precision of the results.
- **target** (*float*) – Target relative to the bounds from 0 (min) to 1 (max) towards which the contrast is reduced

class foolbox.attacks.**LinearSearchContrastReductionAttack**(*, *distance=None*, *steps=1000*, *target=0.5*)

Reduces the contrast of the input using a linear search to find the smallest adversarial perturbation

Parameters

- **distance** (*Optional[foolbox.distances.Distance]*) –
- **steps** (*int*) –
- **target** (*float*) –

class foolbox.attacks.**L2CarliniWagnerAttack**(*binary_search_steps=9*, *steps=10000*, *stepsize=0.01*, *confidence=0*, *initial_const=0.001*, *abort_early=True*)

Implementation of the Carlini & Wagner L2 Attack.[8]

Parameters

- **binary_search_steps** (*int*) – Number of steps to perform in the binary search over the const c.
- **steps** (*int*) – Number of optimization steps within each binary search step.
- **stepsize** (*float*) – Stepsize to update the examples.
- **confidence** (*float*) – Confidence required for an example to be marked as adversarial. Controls the gap between example and decision boundary.
- **initial_const** (*float*) – Initial value of the const c with which the binary search starts.
- **abort_early** (*bool*) – Stop inner search as soons as an adversarial example has been found. Does not affect the binary search over the const c.

[8] Nicholas Carlini, David Wagner, "Towards evaluating the robustness of neural networks. In 2017 ieee symposium on security and privacy" https://arxiv.org/abs/1608.04644

References

class foolbox.attacks.**NewtonFoolAttack**(*steps=100*, *stepsize=0.01*)
Implementation of the NewtonFool Attack.[9]

Parameters

- **steps** (*int*) – Number of update steps to perform.
- **step_size** – Size of each update step.
- **stepsize** (*float*) –

References

class foolbox.attacks.**EADAttack**(*binary_search_steps=9*, *steps=10000*, *initial_stepsize=0.01*, *confidence=0.0*, *initial_const=0.001*, *regularization=0.01*, *decision_rule='EN'*, *abort_early=True*)
Implementation of the EAD Attack with EN Decision Rule.[10]

Parameters

- **binary_search_steps** (*int*) – Number of steps to perform in the binary search over the const c.
- **steps** (*int*) – Number of optimization steps within each binary search step.
- **initial_stepsize** (*float*) – Initial stepsize to update the examples.
- **confidence** (*float*) – Confidence required for an example to be marked as adversarial. Controls the gap between example and decision boundary.
- **initial_const** (*float*) – Initial value of the const c with which the binary search starts.
- **regularization** (*float*) – Controls the L1 regularization.
- **decision_rule** (*Union[typing_extensions.Literal['EN'], typing_extensions.Literal['L1']]*) – Rule according to which the best adversarial examples are selected. They either minimize the L1 or ElasticNet distance.
- **abort_early** (*bool*) – Stop inner search as soons as an adversarial example has been found. Does not affect the binary search over the const c.

References

"EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples", https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewPaper/16893

class foolbox.attacks.**GaussianBlurAttack**(*, *distance=None*, *steps=1000*, *channel_axis=None*, *max_sigma=None*)
Blurs the inputs using a Gaussian filter with linearly increasing standard deviation.

Parameters

- **steps** (*int*) – Number of sigma values tested between 0 and max_sigma.
- **channel_axis** (*Optional[int]*) – Index of the channel axis in the input data.

[9] Uyeong Jang et al., "Objective Metrics and Gradient Descent Algorithms for Adversarial Examples in Machine Learning", https://dl.acm.org/citation.cfm?id=3134635

[10] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh,

- **max_sigma** (*Optional[float]*) – Maximally allowed sigma value of the Gaussian blur.
- **distance** (*Optional[foolbox.distances.Distance]*) –

class foolbox.attacks.**L2DeepFoolAttack**(*, *steps=50*, *candidates=10*, *overshoot=0.02*, *loss='logits'*)

A simple and fast gradient-based adversarial attack.

Implements the DeepFool L2 attack.[11]

Parameters

- **steps** (*int*) – Maximum number of steps to perform.
- **candidates** (*Optional[int]*) – Limit on the number of the most likely classes that should be considered. A small value is usually sufficient and much faster.
- **overshoot** (*float*) – How much to overshoot the boundary.
- **Loss function to use inside the update function.** (*loss*) –
- **typing_extensions.Literal['crossentropy']] loss** (*Union[typing_extensions.Literal['logits'],*) –

References

class foolbox.attacks.**LinfDeepFoolAttack**(*, *steps=50*, *candidates=10*, *overshoot=0.02*, *loss='logits'*)

A simple and fast gradient-based adversarial attack.

Implements the DeepFool L-Infinity attack.

Parameters

- **steps** (*int*) – Maximum number of steps to perform.
- **candidates** (*Optional[int]*) – Limit on the number of the most likely classes that should be considered. A small value is usually sufficient and much faster.
- **overshoot** (*float*) – How much to overshoot the boundary.
- **Loss function to use inside the update function.** (*loss*) –
- **typing_extensions.Literal['crossentropy']] loss** (*Union[typing_extensions.Literal['logits'],*) –

class foolbox.attacks.**SaltAndPepperNoiseAttack**(*steps=1000*, *across_channels=True*, *channel_axis=None*)

Increases the amount of salt and pepper noise until the input is misclassified.

Parameters

- **steps** (*int*) – The number of steps to run.
- **across_channels** (*bool*) – Whether the noise should be the same across all channels.
- **channel_axis** (*Optional[int]*) – The axis across which the noise should be the same (if across_channels is True). If None, will be automatically inferred from the model if possible.

[11] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Pascal Frossard, "DeepFool: a simple and accurate method to fool deep neural networks", https://arxiv.org/abs/1511.04599

class foolbox.attacks.**LinearSearchBlendedUniformNoiseAttack**(*, *distance=None*, *directions=1000*, *steps=1000*)

Blends the input with a uniform noise input until it is misclassified.

Parameters

- **distance** (*Optional[foolbox.distances.Distance]*) – Distance measure for which minimal adversarial examples are searched.
- **directions** (*int*) – Number of random directions in which the perturbation is searched.
- **steps** (*int*) – Number of blending steps between the original image and the random directions.

class foolbox.attacks.**BinarizationRefinementAttack**(*, *distance=None*, *threshold=None*, *included_in='upper'*)

For models that preprocess their inputs by binarizing the inputs, this attack can improve adversarials found by other attacks. It does this by utilizing information about the binarization and mapping values to the corresponding value in the clean input or to the right side of the threshold.

Parameters

- **threshold** (*Optional[float]*) – The threshold used by the models binarization. If none, defaults to (model.bounds()[1] - model.bounds()[0]) / 2.
- **included_in** (*Union[typing_extensions.Literal['lower'], typing_extensions.Literal['upper']]*) – Whether the threshold value itself belongs to the lower or upper interval.
- **distance** (*Optional[foolbox.distances.Distance]*) –

class foolbox.attacks.**DatasetAttack**(*, *distance=None*)

Draws randomly from the given dataset until adversarial examples for all inputs have been found.

To pass data form the dataset to this attack, call `feed()`. `feed()` can be called several times and should only be called with batches that are small enough that they can be passed through the model.

Parameters **distance** (*Optional[foolbox.distances.Distance]*) – Distance measure for which minimal adversarial examples are searched.

class foolbox.attacks.**BoundaryAttack**(*init_attack=None*, *steps=25000*, *spherical_step=0.01*, *source_step=0.01*, *source_step_convergance=1e-07*, *step_adaptation=1.5*, *tensorboard=False*, *update_stats_every_k=10*)

A powerful adversarial attack that requires neither gradients nor probabilities.

This is the reference implementation for the attack.[12]

[12] Wieland Brendel (*), Jonas Rauber (*), Matthias Bethge, "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models", https://arxiv.org/abs/1712.04248

Notes

Differences to the original reference implementation: * We do not perform internal operations with float64 * The samples within a batch can currently influence each other a bit * We don't perform the additional convergence confirmation * The success rate tracking changed a bit * Some other changes due to batching and merged loops

Parameters

- **init_attack** (*Optional[foolbox.attacks.base.MinimizationAttack]*) – Attack to use to find a starting points. Defaults to LinearSearchBlendedUniformNoiseAttack. Only used if starting_points is None.
- **steps** (*int*) – Maximum number of steps to run. Might converge and stop before that.
- **spherical_step** (*float*) – Initial step size for the orthogonal (spherical) step.
- **source_step** (*float*) – Initial step size for the step towards the target.
- **source_step_convergance** (*float*) – Sets the threshold of the stop criterion: if source_step becomes smaller than this value during the attack, the attack has converged and will stop.
- **step_adaptation** (*float*) – Factor by which the step sizes are multiplied or divided.
- **tensorboard** (*Union[typing_extensions.Literal[False], None, str]*) – The log directory for TensorBoard summaries. If False, TensorBoard summaries will be disabled (default). If None, the logdir will be runs/CURRENT_DATETIME_HOSTNAME.
- **update_stats_every_k** (*int*) –

References

class foolbox.attacks.**L0BrendelBethgeAttack**(*init_attack=None*, *overshoot=1.1*, *steps=1000*, *lr=0.001*, *lr_decay=0.5*, *lr_num_decay=20*, *momentum=0.8*, *tensorboard=False*, *binary_search_steps=10*)

L0 variant of the Brendel & Bethge adversarial attack. **[#Bren19]_** This is a powerful gradient-based adversarial attack that follows the adversarial boundary (the boundary between the space of adversarial and non-adversarial images as defined by the adversarial criterion) to find the minimum distance to the clean image.

This is the reference implementation of the Brendel & Bethge attack.

References

Parameters

- **init_attack** (*Optional[foolbox.attacks.base.MinimizationAttack]*) –
- **overshoot** (*float*) –
- **steps** (*int*) –
- **lr** (*float*) –
- **lr_decay** (*float*) –
- **lr_num_decay** (*int*) –
- **momentum** (*float*) –

- **None, str] tensorboard** (*Union[typing_extensions.Literal[False],]*) –
- **binary_search_steps** (*int*) –

class foolbox.attacks.**L1BrendelBethgeAttack**(*init_attack=None*, *overshoot=1.1*, *steps=1000*, *lr=0.001*, *lr_decay=0.5*, *lr_num_decay=20*, *momentum=0.8*, *tensorboard=False*, *binary_search_steps=10*)

L1 variant of the Brendel & Bethge adversarial attack. [#Bren19]_ This is a powerful gradient-based adversarial attack that follows the adversarial boundary (the boundary between the space of adversarial and non-adversarial images as defined by the adversarial criterion) to find the minimum distance to the clean image.

This is the reference implementation of the Brendel & Bethge attack.

References

Parameters

- **init_attack** (*Optional[foolbox.attacks.base.MinimizationAttack]*) –
- **overshoot** (*float*) –
- **steps** (*int*) –
- **lr** (*float*) –
- **lr_decay** (*float*) –
- **lr_num_decay** (*int*) –
- **momentum** (*float*) –
- **None, str] tensorboard** (*Union[typing_extensions.Literal[False],]*) –
- **binary_search_steps** (*int*) –

class foolbox.attacks.**L2BrendelBethgeAttack**(*init_attack=None*, *overshoot=1.1*, *steps=1000*, *lr=0.001*, *lr_decay=0.5*, *lr_num_decay=20*, *momentum=0.8*, *tensorboard=False*, *binary_search_steps=10*)

L2 variant of the Brendel & Bethge adversarial attack. [#Bren19]_ This is a powerful gradient-based adversarial attack that follows the adversarial boundary (the boundary between the space of adversarial and non-adversarial images as defined by the adversarial criterion) to find the minimum distance to the clean image.

This is the reference implementation of the Brendel & Bethge attack.

References

Parameters

- **init_attack** (*Optional[foolbox.attacks.base.MinimizationAttack]*) –
- **overshoot** (*float*) –
- **steps** (*int*) –
- **lr** (*float*) –
- **lr_decay** (*float*) –
- **lr_num_decay** (*int*) –
- **momentum** (*float*) –
- **None, str] tensorboard** (*Union[typing_extensions.Literal[False],*) –
- **binary_search_steps** (*int*) –

class foolbox.attacks.**LinfinityBrendelBethgeAttack**(*init_attack=None*, *overshoot=1.1*, *steps=1000*, *lr=0.001*, *lr_decay=0.5*, *lr_num_decay=20*, *momentum=0.8*, *tensorboard=False*, *binary_search_steps=10*)

L-infinity variant of the Brendel & Bethge adversarial attack. [#Bren19]_ This is a powerful gradient-based adversarial attack that follows the adversarial boundary (the boundary between the space of adversarial and non-adversarial images as defined by the adversarial criterion) to find the minimum distance to the clean image.

This is the reference implementation of the Brendel & Bethge attack.

References

Parameters

- **init_attack** (*Optional[foolbox.attacks.base.MinimizationAttack]*) –
- **overshoot** (*float*) –
- **steps** (*int*) –
- **lr** (*float*) –
- **lr_decay** (*float*) –
- **lr_num_decay** (*int*) –
- **momentum** (*float*) –
- **None, str] tensorboard** (*Union[typing_extensions.Literal[False],*) –
- **binary_search_steps** (*int*) –

foolbox.attacks.**FGM**

alias of foolbox.attacks.fast_gradient_method.L2FastGradientAttack

foolbox.attacks.**FGSM**
alias of foolbox.attacks.fast_gradient_method.LinfFastGradientAttack

foolbox.attacks.**L2PGD**
alias of foolbox.attacks.projected_gradient_descent.L2ProjectedGradientDescentAttack

foolbox.attacks.**LinfPGD**
alias of foolbox.attacks.projected_gradient_descent.LinfProjectedGradientDescentAttack

foolbox.attacks.**PGD**
alias of foolbox.attacks.projected_gradient_descent.LinfProjectedGradientDescentAttack

CHAPTER

THREE

FOOLBOX.CRITERIA

Criteria are used to define which inputs are adversarial. We provide common criteria for untargeted and targeted adversarial attacks, e.g. *Misclassification* and *TargetedMisclassification*. New criteria can easily be implemented by subclassing *Criterion* and implementing *Criterion.__call__()*.

Criteria can be combined using a logical and `criterion1 & criterion2` to create a new criterion.

3.1 Misclassification

```
from foolbox.criteria import Misclassification
criterion = Misclassification(labels)
```

class foolbox.criteria.**Misclassification**(*labels*)

Considers those perturbed inputs adversarial whose predicted class differs from the label.

Parameters **labels** (*Any*) – Tensor with labels of the unperturbed inputs `(batch,)`.

3.2 TargetedMisclassification

```
from foolbox.criteria import TargetedMisclassification
criterion = TargetedMisclassification(target_classes)
```

class foolbox.criteria.**TargetedMisclassification**(*target_classes*)

Considers those perturbed inputs adversarial whose predicted class matches the target class.

Parameters **target_classes** (*Any*) – Tensor with target classes `(batch,)`.

3.3 Criterion

class foolbox.criteria.**Criterion**

Abstract base class to implement new criteria.

abstract **__call__**(*perturbed*, *outputs*)

Returns a boolean tensor indicating which perturbed inputs are adversarial.

Parameters

- **perturbed** (*T*) – Tensor with perturbed inputs `(batch, ...)`.
- **outputs** (*T*) – Tensor with model outputs for the perturbed inputs `(batch, ...)`.

Returns A boolean tensor indicating which perturbed inputs are adversarial `(batch,)`.

Return type T

CHAPTER

FOUR

FOOLBOX.DISTANCES

4.1 Detailed description

class foolbox.distances.**Distance**

class foolbox.distances.**LpDistance**(*p*)

Parameters **p** (*float*) –

clip_perturbation(*references*, *perturbed*, *epsilon*)

Clips the perturbations to epsilon and returns the new perturbed

Parameters

- **references** (*T*) – A batch of reference inputs.
- **perturbed** (*T*) – A batch of perturbed inputs.
- **epsilon** (*float*) –

Returns A tenosr like perturbed but with the perturbation clipped to epsilon.

Return type T

CHAPTER

FIVE

FOOLBOX.UTILS

foolbox.utils.**accuracy**(*fmodel*, *inputs*, *labels*)

Parameters

- **fmodel** (*foolbox.models.base.Model*) –
- **inputs** (*Any*) –
- **labels** (*Any*) –

Return type float

foolbox.utils.**samples**(*fmodel*, *dataset='imagenet'*, *index=0*, *batchsize=1*, *shape=(224*, *224)*, *data_format=None*, *bounds=None*)

Parameters

- **fmodel** (*foolbox.models.base.Model*) –
- **dataset** (*str*) –
- **index** (*int*) –
- **batchsize** (*int*) –
- **int] shape** (*Tuple[int,*) –
- **data_format** (*Optional[str]*) –
- **bounds** (*Optional[foolbox.types.Bounds]*) –

Return type Any

CHAPTER

SIX

FOOLBOX.PLOT

foolbox.plot.**images**(*images*, ***, *n=None*, *data_format=None*, *bounds=(0*, *1)*, *ncols=None*, *nrows=None*, *figsize=None*, *scale=1*, ***kwargs*)

Parameters

- **images** (*Any*) –
- **n** (*Optional[int]*) –
- **data_format** (*Optional[str]*) –
- **float] bounds** (*Tuple[float,*) –
- **ncols** (*Optional[int]*) –
- **nrows** (*Optional[int]*) –
- **float]] figsize** (*Optional[Tuple[float,*) –
- **scale** (*float*) –
- **kwargs** (*Any*) –

Return type None

CHAPTER

SEVEN

FOOLBOX.ZOO

7.1 Get Model

foolbox.zoo.**get_model**(*url*, *module_name='foolbox_model'*, *overwrite=False*, ***kwargs*)

Download a Foolbox-compatible model from the given Git repository URL.

Examples

Instantiate a model:

```
>>> from foolbox import zoo
>>> url = "https://github.com/bveliqi/foolbox-zoo-dummy.git"
>>> model = zoo.get_model(url)
```

Only works with a foolbox-zoo compatible repository. I.e. models need to have a *foolbox_model.py* file with a *create()*-function, which returns a foolbox-wrapped model.

Using the kwargs parameter it is possible to input an arbitrary number of parameters to this methods call. These parameters are forwarded to the instantiated model.

Example repositories:

- https://github.com/jonasrauber/foolbox-tensorflow-keras-applications
- https://github.com/bethgelab/AnalysisBySynthesis
- https://github.com/bethgelab/mnist_challenge
- https://github.com/bethgelab/cifar10_challenge
- https://github.com/bethgelab/convex_adversarial
- https://github.com/wielandbrendel/logit-pairing-foolbox.git
- https://github.com/bethgelab/defensive-distillation.git

Parameters

- **url** (*str*) – URL to the git repository.
- **module_name** (*str*) – The name of the module to import.
- **kwargs** (*Any*) – Optional set of parameters that will be used by the to be instantiated model.
- **overwrite** (*bool*) –

Returns A Foolbox-wrapped model instance.

Return type foolbox.models.base.Model

7.2 Fetch Weights

foolbox.zoo.**fetch_weights**(*weights_uri*, *unzip=False*)

Provides utilities to download and extract packages containing model weights when creating foolbox-zoo compatible repositories, if the weights are not part of the repository itself.

Examples

Download and unzip weights:

```
>>> from foolbox import zoo
>>> url = 'https://github.com/MadryLab/mnist_challenge_models/raw/master/secret.
↪zip'  # noqa F501
>>> weights_path = zoo.fetch_weights(url, unzip=True)
```

Parameters

- **weights_uri** (*str*) – The URI to fetch the weights from.
- **unzip** (*bool*) – Should be *True* if the file to be downloaded is a zipped package.

Returns Local path where the weights have been downloaded and potentially unzipped to.

Return type str

CHAPTER

EIGHT

FOOLBOX.DEVUTILS

Internal module with utility functions

foolbox.devutils.**atleast_kd**(*x*, *k*)

Parameters

- **x** (*eagerpy.Tensor*) –
- **k** (*int*) –

Return type eagerpy.Tensor

foolbox.devutils.**flatten**(*x*, *keep=1*)

Parameters

- **x** (*eagerpy.Tensor*) –
- **keep** (*int*) –

Return type eagerpy.Tensor

CHAPTER

NINE

FOOLBOX.TENSORBOARD

Internal module for attacks that support logging to TensorBoard

class foolbox.tensorboard.**TensorBoard**(*logdir*)

A custom TensorBoard class that accepts EagerPy tensors and that can be disabled by turned into a noop by passing logdir=False.

This makes it possible to add tensorboard logging without any if statements and without any computational overhead if it's disabled.

Parameters **None, str] logdir** (*Union[typing_extensions.Literal[False],*) –

close()

Return type None

conditional_mean(*tag*, *x*, *cond*, *step*)

Parameters

- **tag** (*str*) –
- **x** (*eagerpy.Tensor*) –
- **cond** (*eagerpy.Tensor*) –
- **step** (*int*) –

Return type None

histogram(*tag*, *x*, *step*, *, *first=True*)

Parameters

- **tag** (*str*) –
- **x** (*eagerpy.Tensor*) –
- **step** (*int*) –
- **first** (*bool*) –

Return type None

mean(*tag*, *x*, *step*)

Parameters

- **tag** (*str*) –
- **x** (*eagerpy.Tensor*) –
- **step** (*int*) –

Return type None

probability(*tag*, *x*, *step*)

Parameters

- **tag** (*str*) –
- **x** (*eagerpy.Tensor*) –
- **step** (*int*) –

Return type None

probability_ratio(*tag*, *x*, *y*, *step*)

Parameters

- **tag** (*str*) –
- **x** (*eagerpy.Tensor*) –
- **y** (*eagerpy.Tensor*) –
- **step** (*int*) –

Return type None

scalar(*tag*, *x*, *step*)

Parameters

- **tag** (*str*) –
- **float] x** (*Union[int,*) –
- **step** (*int*) –

Return type None

foolbox.tensorboard.**maybenoop**(*f*)

Parameters **f** (*F*) –

Return type F

CHAPTER

TEN

FOOLBOX.TYPES

class foolbox.types.**Bounds**(*lower*, *upper*)

Parameters

- **lower** (*float*) –
- **upper** (*float*) –

property lower
Alias for field number 0

property upper
Alias for field number 1

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

P

S

T

U

V